USB2 Loopback Plug
Application Programming Interface
User Guide

Version: 2.3
Date: 23 January 2025
Web site: https://www.passmark.com/products/usb2api/

Table of Contents

Introduction....3
System Requirements....3
Usage....3
Data Types....4
Testing Modes....4
Test Pattern Types....4
Array Size....4
Function Definitions....5
USB2NumPlugsConnected....5
USB2GetConnectedPlugs....5
USB2GetUSBEnumerationType....6
USB2GgetDeviceInfo....6
USB2SetTestParameters....7
USB2StartTest....8
USB2StopTest....8
USB2TestIsRunning....8
USB2GetLoopbackResults....9
USB2GetBenchmarkResults....9
USB2GetBusErrors....10
USB2GetLastError....10
USB2GetUSBErrors....11

Introduction

The USB2 Application Programming Interface has been designed to allow access by 3rd party programs to the PassMark USB2 loop back plug (https://www.passmark.com/products/usb2loopback). It provides several functions to communicate with the USB2 loop back plugs and an easy way to use the plugs without having to write any device driver or extra firmware code.

The usb2api.zip file contains:

```
bin\Win32          USB2Dll.lib USB2Dll.dll API 32-bit library files
bin\x64            USB2Dll.lib USB2Dll.dll API 64-bit library files
doc\               This help document
include\           USB2DLL.h API header file
USB2_DLL_Demo\     Source code for the demo application
```

System Requirements

The API has been designed and tested to run on Windows XP SP3, 2003 server, Vista, Windows 7, Windows 8, 2008 Server, Windows 10 and Windows 11. Other operating systems are not supported.

Usage

There are three files that make up the API, USB2Dll.lib, USB2Dll.dll and USB2DLL.h.

To use the DLL, these steps need to be followed;

- Add the USB2DLL.h file to the project and #include it where necessary
- Add the appropriate 32-bit or 64-bit USB2Dll.lib to the project settings
- Copy the appropriate 32-bit or 64-bit USB2Dll.dll to the same directory as the executable that will be calling the API functions

Data Types

Testing Modes

```
#define LOOPBACK    1
#define BENCHMARK 2
```

When making a call to USB2SetTestParameters use one of these values to define the test type.

Test Pattern Types

```
typedef enum _DATA_PATTERN
{
        INCREMENTINGBYTE,
        RANDOMBYTE,
        INCREMENTINGDWORD,
        CONSTANTBYTE
} DATA_PATTERN;
```

INCREMENTINGBYTE	Starts at a preset value and increases by 1 per byte sent
INCREMENTINGWORD	Starts at a preset value and increases by 1 per 4 bytes sent.
RANDOMBYTE	Uses a randomly generated value for each byte sent
CONSTANTBYTE	Uses a preset value for each byte sent

INCREMENTINGWORD is only valid for benchmark testing.

When making a call to USB2SetTestParameters use one of these values to define the test pattern type.

Array Size

```
#define MAX_USB_DEV_NUMBER 32
```

Most arrays that are passed to functions in the USB2 API dealing with the connected USB2 plugs are expected to be this size.

Function Definitions

USB2NumPlugsConnected

int USB2NumPlugsConnected ()

Description

Get the number of PassMark USB2Loopback plugs currently connected (with device driver loaded) to the system.

Return Values

The number of PassMark USB2Loopback plugs currently connected (with device driver loaded) to the system.

USB2GetConnectedPlugs

bool USB2GetConnectedPlugs (bool * bDeviceArray)

Description

Get an array of USB Device numbers that identify each USB2 plug currently connected to the system. The device number is assigned by the USB2 device driver and is the key when accessing the USB2Loopback device.
The device number is assigned on connection, and de-assigned on de-connection. The plug may receive a different number when reconnecting to the system.

Parameters

bDeviceArray	[out] Pointer to array indicating if a plug is connected at the device number indicated by the index. The array must be of size: MAX_USB_DEV_NUMBER

Return Values

True for success. False for failure.

USB2GetUSBEnumerationType

bool USB2GetUSBEnumerationType (int iDeviceNum, bool * bHighSpeed,
int * iFirmWare)

Description

Get info about the USB plug, whether it has connected as USB 1 or 2 and the firmware version

Parameters

iDeviceNum	[in] Number referring to the device to query (index in the connected array, see USB2GetConnectedPlugs)
bHighSpeed	[out] Value changed to indicate USB 1/2 status, true for high speed
iFirmWare	[out] Value changed to hold the firmware version

Return Values

True for success. False for failure.

USB2GgetDeviceInfo

int USB2GetDeviceInfo (int PortNum, char *Serial, int MaxSerialLen, char *Desc,
int MaxDescLen)

Description

Gets the serial number and the device description for the USB device identified by Index.

Parameters

PortNum	[in] Number referring to the device to query (index in the connected array, see USB2GetConnectedPlugs)
MaxSerialLen	[in] Max length of buffer for serial num
MaxDescLen	[in] Max length of buffer for description
Serial	[out] Device serial number string
Desc	[out] Device description string

Return Values

1 if OK
2 if OK, but EEPROM on device doesn't look to be programmed correctly
0 if error

***Please note this function has changed in version 1.5, duration is now seconds instead of minutes.**

USB2SetTestParameters

```
bool USB2SetTestParameters (bool useplug[MAX_USB_DEV_NUMBER],
                            int mode, unsigned int duration,
                            DATA_PATTERN dataPattern,
                            unsigned long patternStart, bool verifyData)
```

Description

Set which test to run, the test parameters and which plugs to use. See the Data Types sections above for the expected mode and dataPattern values.

Parameters

useplug	[in] Array with the values for the plugs to test set to true
mode	[in] Test mode (benchmark/loopback)
duration	[in] Test duration in seconds
dataPattern	[in] Type of test data (constant, incrementing, random)
patternStart	[in] Start value for the test pattern. INCREMENTINGBYTE will cast the value to a single byte and INCREMENTINGDWORD will use all 4 bytes of this value and will increment during the test. CONSTANTBYTE casts the value to a single byte which it repeats for the entire test.
verifyData	[in] True to verify data

Return Values

True if successful.
False if test parameters are incompatible, eg trying to benchmark more than 1 plug at a time. Call USB2GetLastError to get the error string.

USB2StartTest

bool USB2StartTest ()

Description

Start running the tests

Return Values

True on success, False on failure. Call USB2GetLastError to get the error string.

USB2StopTest

bool USB2StopTest ()

Description

Stops any running tests

Return Values

Returns true

USB2TestIsRunning

bool USB2TestIsRunning ()

Description

Call this function to determine if there are tests still running.

Return Values

Returns true is USB2 tests are still running, false if they has finished

USB2GetLoopbackResults

USB2GetLoopbackResults (int iOperations[MAX_USB_DEV_NUMBER],
int iErrors[MAX_USB_DEV_NUMBER])

Description

Get the current results for any running loopback tests. Call USB2GetUSBErrors to retrieve any errors logged during the test.

Parameters

iOperations	[out] Array of size MAX_USB_DEV_NUMBER with the number of operations for each plug being tested
iErrors	[out] Array of size MAX_USB_DEV_NUMBER with the number of errors for each plug being tested

Return Values

True if successful, False on failure. Call USB2GetLastError to get the error string.

USB2GetBenchmarkResults

bool USB2GetBenchmarkResults (int* iOperations, int* iErrors, float* maxRate,
float* maxReadRate, float* maxWriteRate,
float* avgRate, float* avgPossible);

Description

Get the current results for the running benchmark test, the data rates returned are in Megabits/s.

Note: You can only benchmark one USB port at a time (unlike the loopback test). To set the device number of the port to be benchmarked, use USB2SetTestParameters() to specify a single index in the “useplug” array to the value true. Call USB2GetUSBErrors to retrieve any errors logged during the test.

Parameters

iOperations	[out] The number of operations for the plug being tested
iErrors	[out] The number of errors for the plug being tested
maxRate	[out] The max transfer rate for the plug being tested
maxReadRate	[out] The max read rate for the plug being tested
maxWriteRate	[out] The max write rate for the plug being tested
avgRate	[out] The average transfer rate for the plug being tested
avgPossible	[out] The average possible transfer rate for the plug being tested

Return Values

True if successful, False for failure. Call USB2GetLastError to get the error string.

USB2GetBusErrors

int USB2GetBusErrors(int iDeviceNum)

Description

Gets the number of bus errors for the specified USB2 plug

Parameters

iDeviceNum	[in] Number referring to the USB2 device to query (index in the connected array)

Return Values

The number of bus errors

USB2GetLastError

bool USB2GetLastError(char* msg, int* msgSize)

Description

Return the error string that was last set. After the error message is successfully copied, the error string is cleared. This is set when one of the USB2 API function calls fails for any reason.

Parameters

msg	[out] The last error message is copied into this buffer
msgSize	[in/out] This is the max size of the passed buffer. I the error message is too big, the required size will be returned in this variable and the function will return false

Return Values

True if successful. False if supplied buffer is too small and required buffer size will be returned in msgSize.

USB2GetUSBErrors

```
bool USB2GetUSBErrors(
        char errorLog[USB_ERRORLOG_MAX][USB_ERRORLOG_MSG_SIZE],
        int* numCopied, bool clearLog)
```

Description

Return the last 20 (USB_ERRORLOG_MAX) USB errors that have been logged during the test. The log is only cleared if specified by the caller.

Parameters

errorLog	[out] On return will contain the last logged USB errors
numCopied	[out] The number of error messages that were copied
clearLog	[in] When true clear the log after copying

Return Values

True if successful, False on failure.